Hamiltonian System Approach to Distributed Spectral Decomposition in Networks

Konstantin Avrachenkov, Philippe Jacquet, Jithin Sreedharan

Hamiltonian System Approach to Distributed Spectral Decomposition in Networks

Konstantin Avrachenkov
INRIA, France
Email: k.avrachenkov@inria.fr

Philippe Jacquet
Nokia Bell Labs, France
Email: philippe.jacquet@nokia-bell-labs.com

Jithin K. Sreedharan
Purdue University, USA
Email: jithinks@purdue.edu

***Abstract*—Because of the significant increase in size and complexity of the networks, the distributed computation of eigenvalues and eigenvectors of graph matrices has become very challenging and yet it remains as important as before. In this paper we develop efficient distributed algorithms to detect, with higher resolution, closely situated eigenvalues and corresponding eigenvectors of symmetric graph matrices. We model the system of graph spectral computation as physical systems with Lagrangian and Hamiltonian dynamics. The spectrum of Laplacian matrix, in particular, is framed as a classical spring-mass system with Lagrangian dynamics. The spectrum of any general symmetric graph matrix turns out to have a simple connection with quantum systems and it can be thus formulated as a solution to a Schrödinger-type differential equation. Taking into account the higher resolution requirement in the spectrum computation and the related stability issues in the numerical solution of the underlying differential equation, we propose the application of symplectic integrators to the calculation of eigenspectrum. The effectiveness of the proposed techniques is demonstrated with numerical simulations on real-world networks of different sizes and complexities.**

I. Introduction

Consider an undirected graph $G = (V, E)$ with V as the vertex set ($n := |V|$) and E as the edge set ($m := |E|$). Let M be any symmetric matrix associated with G. Broadly speaking, we say that M is a graph matrix if it has non-zero elements on the edge set. Due to symmetry, the eigenvalues of M are real and can be ranked in ascending order as $\lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_n$. We investigate efficient techniques to compute the eigenvalues $\lambda_1, \ldots, \lambda_n$ and the corresponding eigenvectors $\boldsymbol{u}_1, \ldots, \boldsymbol{u}_n$, in a distributed way.

We define two typical matrices which appear frequently in network analysis. First one is the adjacency matrix $A \in \mathbb{R}^{n \times n}$ in which the individual entries are given by

$$a_{uv} = \begin{cases} 1 & \text{if } u \text{ is a neighbour of } v, \\ 0 & \text{otherwise.} \end{cases}$$

Since the focus in this paper is on undirected graphs, $A^\intercal = A$. The matrix A is also called the unweighted adjacency matrix and one can also define a weighted version in which the weight 1 for an edge is replaced by any weight such that $a_{uv} = a_{vu}$. Another matrix which is found very common in many graph theoretic problems is the Laplacian matrix $L = [\ell_{i,j}] := D - A$. Here the matrix D is a diagonal matrix with diagonal elements equal to degrees of the nodes $d_1, \ldots, d_n$.

A. Applications of graph spectrum

The knowledge of $\{\lambda_i\}$'s and $\{\boldsymbol{u}_i\}$'s can be made use of in many ways. For instance, spectral clustering is a prominent solution which exploits the first k eigenvectors of the Laplacian matrix for identifying the clusters in a network [1]. Another classical use of Laplacian eigenvalues is in computing the number of spanning trees of a graph G which is $n^{-1}\,\lambda_2\lambda_3 \ldots \lambda_n$. Many studies have been conducted on the use of the spectrum of the adjacency matrix such as computing the number of traingles of a netwok (both locally and globally) [2], graph dimensionality reduction and link reduction [3] etc.

Two applications relevant to multi-agent and multi-dimensional systems will be explained in detail in Section VI.

B. Our basic approach

Let M be any symmetric graph matrix. Consider the following Schrödinger-type differential equation

$$\frac{\partial}{\partial t}\boldsymbol{\psi}(t) = iM\,\boldsymbol{\psi}(t), \tag{1}$$

where $\boldsymbol{\psi}(t)$ is a complex valued n dimension vector, which can be interpreted as the wave function of a hypothetical quantum system. The solution of this differential equation with the boundary condition $\boldsymbol{\psi}(0) = \boldsymbol{a}_0$ is $\exp(iMt)\boldsymbol{a}_0$. Subjecting this solution to the Fourier transform provides a decomposition in terms of eigenvalues and eigenvectors as follows:

$$\int_{-\infty}^{+\infty} e^{iMt}\boldsymbol{a}_0 e^{-it\theta}dt = 2\pi \sum_{j=1}^{n} \delta_{\lambda_j}(\theta)\boldsymbol{u}_j(\boldsymbol{u}_j^\intercal \boldsymbol{a}_0), \tag{2}$$

where δ_{λ_j} is the Dirac function shifted by λ_j. This follows from the eigen-decomposition of the matrix M, $e^{iMt} = \sum_j e^{it\lambda_j}\boldsymbol{u}_j\boldsymbol{u}_j^\intercal$. In order to avoid the harmonic oscillations created from finite and discretized version of the above Fourier transform, which will mask the Dirac peaks, the following Gaussian smoothing can be performed. For $v > 0$,

$$\begin{aligned} &\frac{1}{2\pi}\int_{-\infty}^{+\infty} e^{iMt}\boldsymbol{a}_0 e^{-t^2 v/2} e^{-it\theta}dt \\ &\quad = \sum_{j=1}^{n} \frac{1}{\sqrt{2\pi v}} \exp(-\frac{(\lambda_j - \theta)^2}{2v})\boldsymbol{u}_j(\boldsymbol{u}_j^\intercal \boldsymbol{a}_0). \end{aligned} \tag{3}$$

The right-hand side of the above expression leads us to a plot at each node k with Gaussian peaks at each of the eigenvalues and the amplitude of the peak at jth eigenvalue

is $(\sqrt{2\pi v})^{-1}(\boldsymbol{u}_j^\intercal \boldsymbol{a}_0)\boldsymbol{u}_j(k)$, proportional to the kth component in the eigenvector $\boldsymbol{u}_j$.

The idea is to estimate the solution of (1) at ε intervals of time for a total of s samples. One way to form an estimate to the left-hand side to (3) is:

$$\varepsilon\Re\Big(\boldsymbol{b}_0 + 2\sum_{\ell=1}^{s} e^{i\varepsilon\ell M}\boldsymbol{b}_0 e^{-i\ell\varepsilon\theta}e^{-\ell^2\varepsilon^2 v/2}\Big). \tag{4}$$

In [4], we use the above approximation with various approaches based on diffusion, gossiping and quantum random walks for distributed computation of the eigenvalues and the eigenvectors. Let us discuss some challenges in this approach.

Issues in the computation

While doing numerical experiments, we have observed that the approaches in [4] work well for larger eigenvalues of the adjacency matrix of a graph but they do not perform that well when one needs to distinguish between the eigenvalues which are very close to each other. One of the main techniques proposed there to solve (1) and to find $e^{i\varepsilon\ell M}$ in (4) are via rth order Runge-Kutta method and its implementation as a diffusion process in the network. The r-th order Runge-Kutta method has the convergence rate of $\mathcal{O}(\varepsilon^r)$. We have observed that this is the case while checking the trajectory of the associated differential equation solution; the solution diverges, and it happens when a large number of iterations s is required (see Section V).

A larger value for s is anticipated from our approximation in (4) due to the following facts. From the theory of Fourier transform and Nyquist sampling, the following conditions must be satisfied:

$$\varepsilon \leq \frac{\pi}{\lambda_n} \text{ and } s \geq \frac{2\pi}{\varepsilon\lambda_{\text{diff}}}, \tag{5}$$

where λ_{diff} is the maximum resolution we require in the frequency (eigenvalue) domain, which is ideally $\min_i |\lambda_i - \lambda_{i+1}|$. This explains that when dealing with graph matrices with larger λ_n and require higher resolution, s will take large values. For instance, in case of the Laplacian matrix, where the maximum eigenvalue is bounded as $\frac{n}{n-1}\Delta(G) \leq \lambda_n \leq 2\Delta(G)$, with $\Delta(G)$ as the maximum degree of the graph and the lower eigenvalues are very close to each other, s turns out to be typically a very large value.

In what follows, we propose solutions to the above mentioned issues.

C. Related works

The general idea of using mechanical oscillatory behaviour for the detection of eigenvalues has appeared in a few previous works, see e.g., [5], [6]. Though the technique in [5] is close to ours, our methods differ by focussing on a Schrödinger-type equation and numerical integrators specific to it. Moreover, we demonstrate the efficiency and stability of the methods in real-world networks of varying sizes, in contrast to a small size synthetic network considered in [5], and our methods can be used to estimate eigenvectors as well.

In comparison to [6] we do not deform the system and we use new symplectic numerical integrators [7], [8]. For the problem of distributed spectral decomposition in networks, one of the first and prominent works appeared in [3]. But their algorithm requires distributed orthonormalization at each step and they solve this difficult operation via random walks. But if the graph is not well-connected (low conductance), this task will take a very long time to converge. Our distributed implementation based on fluid diffusion in the network does not require such orthonormalization.

D. Contributions

We make the following contributions and significantly improve the algorithms from our previous work:

1) We observe from our previous studies that the stability in trajectory of the differential equation solver is of significant influence in the eigenvalue-eigenvector technique based on (1). Thus, we resort to geometric integrators to ensure the stability. In particular, by modeling the problem as a Hamiltonian system, we use symplectic integrators (SI) which protect the volume preservation of Hamiltonian dynamics, thus preserve stability and improve accuracy.
2) We propose algorithms that are easy to design without involving many parameters with interdependence, compared to the algorithms proposed in [4].

In the rest of the paper for clarity of presentation we mostly concentrate on the Laplacian matrix L as an example for graph matrix M. We design algorithms based on Lagrangian as well as Hamiltonian dynamics, to compute the smallest k eigenvalues and the respective eigenvectors of the Laplacian matrix efficiently. For simplicity, in this paper we do not consider Gaussian smoothing (3), but the proposed algorithms can be readily extended to include it.

The paper is organized as follows. In Section II, we explain a mass-spring analogy specific to Laplacian matrix and derive a method to identify the spectrum. Section III focuses on general symmetric matrices and develop techniques based on solving the Schrödinger-type equation efficiently. Section IV details a distributed implementation of the proposed algorithm. Section V contains numerical simulations on networks of different sizes. Section VI contains two relevant applications to multi-agent and multi-dimensional systems. Section VII concludes the paper.

For convenience we summarize the important notation used in this paper in Table I.

II. Mechanical spring analogy with Lagrangian dynamics

Consider a hypothetical mechanical system representation of the graph G in which unit masses are placed on the vertices and the edges are replaced with mechanical springs of unit stiffness. Using either Lagrangian or Netwonian mechanics, the dynamics of this system is described by the following system of differential equations

$$\ddot{\boldsymbol{x}}(t) + L\boldsymbol{x}(t) = \boldsymbol{0}. \tag{6}$$

Notation	Meaning
G, (V, E)	Graph, Node set and edge set
n, m	No. of nodes, no. of edges
A	Adjacency matrix
L	Laplacian matrix
$\lambda_1, \ldots, \lambda_k$	Smallest k eigenvalues of L in ascending order
$\boldsymbol{u}_1, \ldots, \boldsymbol{u}_k$	Eigenvectors corresponding to $\lambda_1, \ldots, \lambda_k$
d_j	Degree of node j without including self loop
$\Delta(G)$	$\max\{d_1, \ldots, d_n\}$
$\mathcal{N}(m)$	Neighbor list of node m without including self loop
ε	Sampling interval in time domain
T, s	Total time frame and no. of samples $\lceil T/\varepsilon \rceil$
$\boldsymbol{x}(t), \boldsymbol{x}_i$	vector $\boldsymbol{x}$ with index in continuous and discrete time
$\boldsymbol{x}_i[k]$	kth component of a vector $\boldsymbol{x}_i$

TABLE I: List of important notations

The system has the Hamiltonian function as $\mathcal{H} = \frac{1}{2}\dot{\boldsymbol{x}}^\intercal I \dot{\boldsymbol{x}} + \frac{1}{2}\boldsymbol{x}^\intercal L \boldsymbol{x}$, I being the identity matrix of order n.

We note that once we obtain by some identification method the frequencies ω_k of the above oscillatory system, the eigenvalues of the Laplacian L can be immediately retrieved by the simple formula $\lambda_k = |\omega_k^2|$. This will be made clearer later in this section.

Starting with a random initial vector $\boldsymbol{x}(0)$, we can simulate the motion of this spring system. For the numerical integration, the Leapfrog or Verlet method [9] technique can be applied. Being an example of geometric integrator, Verlet method has several remarkable properties. It has the same computational complexity as the Euler method but it is of second order method (Euler method is employed in [4] as the first order distributed diffusion). In addition, the Verlet method is stable for oscillatory motion and conserves the errors in energy and computations [10, Chapter 4]. It has the following two forms. Let $\boldsymbol{p}(t) := \dot{\boldsymbol{x}}(t)$ and $\boldsymbol{x}_i$ be the approximation of $\boldsymbol{x}(i\varepsilon)$, similarly $\boldsymbol{p}_i$ for $\boldsymbol{p}(i\varepsilon)$. Here ε is the step size for integration. First, define

$$\boldsymbol{p}_{1/2} = \boldsymbol{p}_0 + \varepsilon/2(-L\boldsymbol{x}_0).$$

Then, perform the following iterations

$$\begin{aligned} \boldsymbol{x}_i &= \boldsymbol{x}_{i-1} + \varepsilon \boldsymbol{p}_{i-1/2} \\ \boldsymbol{p}_{i+1/2} &= \boldsymbol{p}_{i-1/2} + \varepsilon(-L\boldsymbol{x}_i). \end{aligned}$$

Equivalently, one can do the updates as

$$\begin{aligned} \boldsymbol{x}_{i+1} &= \boldsymbol{x}_i + \varepsilon p_i + \varepsilon^2/2(-L\boldsymbol{x}_i) \\ \boldsymbol{p}_{i+1} &= \boldsymbol{p}_i + \varepsilon[(-L\boldsymbol{x}_i) + (-L\boldsymbol{x}_{i+1})]. \end{aligned}$$

We name the above algorithm as Order-2 Leapfrog.

Solution of the differential equation (6) subject to the boundary values $\boldsymbol{x}(0) = \boldsymbol{a}_0$ and $\boldsymbol{p}(0) = \boldsymbol{b}_0$ is

$$\boldsymbol{x}(t) = \left(\frac{1}{2}\boldsymbol{a}_0 - i\frac{\boldsymbol{b}_0}{\sqrt{\Lambda}}\right) e^{it\sqrt{L}} + \left(\frac{1}{2}\boldsymbol{a}_0 + i\frac{\boldsymbol{b}_0}{\sqrt{\Lambda}}\right) e^{-it\sqrt{L}},$$

where we assume the decomposition of L based on spectral theorem, i.e., $L = U\Lambda U^\intercal$ with U as the orthonormal matrix with columns as eigenvectors and Λ as the diagonal matrix formed from the eigenvalues. Further simplification of the above expression along the fact that $f(L) = Uf(\Lambda)U^\intercal$, for any function f which can be expressed in terms of power series, gives

$$\boldsymbol{x}(t) = \cos(t\sqrt{L})\boldsymbol{a}_0 + (\sqrt{L})^{-1}\sin(t\sqrt{L})\boldsymbol{b}_0.$$

or kth component of $\boldsymbol{x}(t)$ is

$$\boldsymbol{a}_0[k]\cos(t\sqrt{\lambda_k}) + \frac{\boldsymbol{b}_0[k]}{\sqrt{\lambda_k}}\sin(t\sqrt{\lambda_k}).$$

Now we have

$$\begin{aligned} &\int_{-\infty}^{+\infty} \boldsymbol{x}(t)e^{-it\theta}dt \\ &= \int_{-\infty}^{+\infty} \sum_{k=1}^{n} \cos(t\sqrt{\lambda_k})\boldsymbol{u}_k(\boldsymbol{u}_k^\intercal \boldsymbol{a}_0)e^{-it\theta}dt \\ &\quad + \int_{-\infty}^{+\infty} (\sqrt{L})^{-1} \sum_{k=1}^{n} \sin(t\sqrt{\lambda_k})\boldsymbol{u}_k(\boldsymbol{u}_k^\intercal \boldsymbol{b}_0)e^{-it\theta}dt \\ &= \sum_{k=1}^{n} \boldsymbol{u}_k(\boldsymbol{u}_k^\intercal \boldsymbol{a}_0)\left(\pi[\delta(\theta - \sqrt{\lambda_k}) + \delta(\theta + \sqrt{\lambda_k})]\right) \\ &\quad + (\sqrt{L})^{-1}\boldsymbol{u}_k(\boldsymbol{u}_k^\intercal \boldsymbol{b}_0)\left(-\pi i[\delta(\theta - \sqrt{\lambda_k}) - \delta(\theta + \sqrt{\lambda_k})]\right). \end{aligned}$$

Taking the real and positive spectrum will give $\pi \sum_{k=1}^{n} \boldsymbol{u}_k(\boldsymbol{u}_k^\intercal \boldsymbol{a}_0)\delta(\theta - \sqrt{\lambda_k})$. The whole operation can be approximated by applying an s-point FFT on $\{\boldsymbol{x}_i, 0 \le i < s\}$, and taking real values. (To be exact, there is a phase factor to be multiplied to the kth point in FFT approximation, and is given by $(\sqrt{2\pi})^{-1}\varepsilon \exp(-it_0 k\lambda_{\text{diff}})$, where we considered the time interval $[t_0, t_0 + s\varepsilon]$).

Note that (6) is different from the original differential equation in [4] where it is $\dot{\boldsymbol{x}}(t) = iL\boldsymbol{x}(t)$ containing complex coefficients.

III. Hamiltonian Dynamics and Relation with Quantum Random Walk

In [4] we have studied the Schrödinger-type equation of the form (1) with M taken as the adjacency matrix A and $\boldsymbol{\psi}$ as the wave function. Now let us consider a similar equation with respect to the graph Laplacian

$$\dot{\boldsymbol{\psi}}(t) = iL\boldsymbol{\psi}(t). \tag{7}$$

The solution of this dynamics is closely related to the evolution of continuous time quantum random walk and algorithms are developed in [4] based on this observation.

Now since the matrix L is real and symmetric, it is sufficient to use the real-imaginary representation of the wave function $\boldsymbol{\psi}(t) = \boldsymbol{x}(t) + i\boldsymbol{y}(t)$, $\boldsymbol{x}(t), \boldsymbol{y}(t) \in \mathbb{R}$. Substituting this representation into equation (7) and taking real and imaginary parts, we obtain the following system of equations

$$\begin{aligned} \dot{\boldsymbol{x}}(t) &= -L\boldsymbol{y}(t), \\ \dot{\boldsymbol{y}}(t) &= L\boldsymbol{x}(t), \end{aligned}$$

or equivalently in the matrix form

$$\frac{d}{dt}\begin{bmatrix} \boldsymbol{x}(t) \\ \boldsymbol{y}(t) \end{bmatrix} = \begin{bmatrix} 0 & -L \\ L & 0 \end{bmatrix}\begin{bmatrix} \boldsymbol{x}(t) \\ \boldsymbol{y}(t) \end{bmatrix}. \tag{8}$$

Such a system has the following Hamiltonian function

$$\mathcal{H} = \frac{1}{2}\boldsymbol{x}^\intercal L\boldsymbol{x} + \frac{1}{2}\boldsymbol{y}^\intercal L\boldsymbol{y}. \tag{9}$$

Next, the very helpful decomposition

$$\begin{bmatrix} 0 & -L \\ L & 0 \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ L & 0 \end{bmatrix} + \begin{bmatrix} 0 & -L \\ 0 & 0 \end{bmatrix},$$

together with the observation that

$$\exp\left(\begin{bmatrix} 0 & 0 \\ L & 0 \end{bmatrix}\right) = \begin{bmatrix} I & 0 \\ L & I \end{bmatrix},$$

leads us to another modification of the leapfrog method known as symplectic split operator algorithm [7]: Initialize with

$$\boldsymbol{\delta y} = -L\boldsymbol{x}_0,$$

then perform the iterations

$$\boldsymbol{y}_{i-1/2} = \boldsymbol{y}_{i-1} - \frac{\varepsilon}{2}\boldsymbol{\delta y}, \tag{10}$$

$$\boldsymbol{x}_i = \boldsymbol{x}_{i-1} - \varepsilon L\boldsymbol{y}_{i-1/2}, \tag{11}$$

and update

$$\boldsymbol{\delta y} = -L\boldsymbol{x}_i, \tag{12}$$

$$\boldsymbol{y}_i = \boldsymbol{y}_{i-1/2} - \frac{\varepsilon}{2}\boldsymbol{\delta y}. \tag{13}$$

The above modified leapfrog method belongs to the class of symplectic integrator (SI) methods [8], [10], [11]. We name the above algorithm as *order-2 SI*.

The Hamiltonian system approach can be implemented in two ways:

1) Form the complex vector $\boldsymbol{x}_k + i\boldsymbol{y}_k$ at each of the ε intervals. Then $\{\boldsymbol{x}_k + i\boldsymbol{y}_k, 0 \le k < s\}$ with $\boldsymbol{x}_0 = \boldsymbol{a}_0$ and $\boldsymbol{y}_0 = \boldsymbol{b}_0$ approximates $\exp(iLt)(\boldsymbol{a}_0 + i\boldsymbol{b}_0)$ at $t = 0, \varepsilon, \ldots, (s-1)\varepsilon$ intervals. A direct application of s point FFT with appropriate scaling will give the spectral decomposition as in (2).
2) Note that the formulation in (8) is equivalent to the following differential equations

$$\ddot{\boldsymbol{y}}(t) + L^2\boldsymbol{y}(t) = 0, \quad \ddot{\boldsymbol{x}}(t) + L^2\boldsymbol{x}(t) = 0, \tag{14}$$

which are similar to the one in (6) except the term L^2. Now on the same lines of analysis as in the previous section, taking the real and positive spectrum of just $\boldsymbol{y}$ component will give $\pi\sum_{k=1}^{n}\boldsymbol{u}_k(\boldsymbol{u}_k^\intercal\boldsymbol{a}_0)\delta(\theta-\lambda_k)$.

A. Fourth order integrator

The Hamiltonian $\mathcal{H}$ in (9) associated to the Schrödinger-type equation has a special characteristic that it is seperable into two quadratic forms, which help to develop higher order integrators. The r stage integrator has the following form. Between t and $t+\varepsilon$ intervals, we run for $j = 1, \ldots, r$,

$$\boldsymbol{y}_j = \boldsymbol{y}_{j-1} + p_j\varepsilon L\boldsymbol{x}_{j-1}$$

$$\boldsymbol{x}_j = \boldsymbol{x}_{j-1} - q_j\varepsilon L\boldsymbol{y}_j.$$

In order to make qth order integrator $r \le q$. For our numerical studies we take the optimized coefficients for order-4 derived in [12]. We call such algorithm as *order-4 SI*.

IV. Distributed Implementation

The order-2 symplectic integrator algorithm in (10)-(13) can be implemented in a distributed fashion such that each node needs to communicate only to its neighbors. The matrix-vector multiplications in (11) and (12), during one iteration of the algorithm, require diffusion of packets or fluids to the neighbors of every node, and fusion of the received fluids from all the neighbors at each node. Each iteration of the algorithm subsequently has two diffusion-fusion cycles and three synchronization points. A diffusion-fusion cycle consists of $|E|$ packets sent in parallel and hence total number of packets exchanged in one iteration of the algorithm is $2|E|$. Since order-2 SI does not require orthonormalization (unlike classical power iteration and inverse iteration methods for computing eigenelements), and the diffusion-fusion cycle is within one hop neighborhood, time delay of the algorithm will not be too significant. The synchronization points definitely pose some constraints, and demand extra resources. We have also considered an asynchronous version of the distributed algorithm, which will be presented in the extended version of the work.

V. Numerical Results

The parameters ε and s are chosen in the numerical studies satisfying the constraints in (5). We assume that the maximum degree is known to us.

Note that if the only purpose is to detect eigenvalues, not to compute the eigenvectors, then instead of taking real part of the FFT in the Hamiltonian solution, it is clearly better to compute the absolute value of the complex quantity to get higher peaks. But in the following simulations we look for eigenvectors as well.

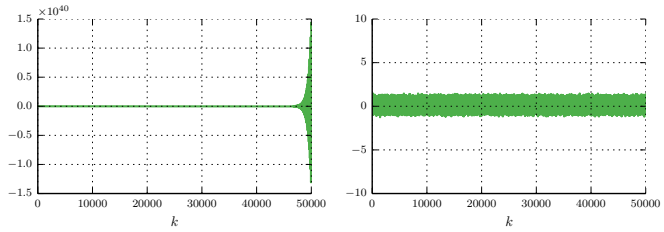
For the numerical studies, in order to show the effectiveness of the distributed implementation, we focus on one particular node and plot the spectrum observed at that node. In the plots, $f_\theta(k)$ indicates the approximated spectrum at frequency θ observed on node k.

A. Les Misérables network

In Les Misérables network, nodes are the characters in the well-known novel with the same name and edges are formed if two characters appear in the same chapter. The number of nodes is 77 and number of edges is 254. We look for the spectral plot at a specific node called Valjean (with node ID 11), a character in the associated novel.

The instability of the Euler method is clear from Figure 1a, whereas Figure 1b shows the guaranteed stability of Hamiltonian SI. Here the y-axis represents the absolute value of $\psi(t)$. (Note the difference in the y-axis scale in the figures). Figure 2 shows the result given by the Lagrangian Leapfrog method from Section II. It can be observed that very few smallest eigenvalues are detected using order-2 Leapfrog compared to the SI technique (order-2) in Figure 3. This demonstrates the superiority of the Hamiltonian system approach. Figure 4 shows order-4 SI with much less number of iterations. The

precision in order-4 plot can be significantly improved further by increasing the number of iterations.



(a) Euler method (b) Hamiltonian order-2 SI

Fig. 1: Trajectories

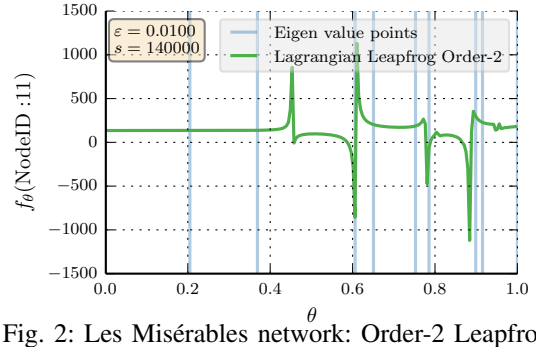


Fig. 2: Les Misérables network: Order-2 Leapfrog

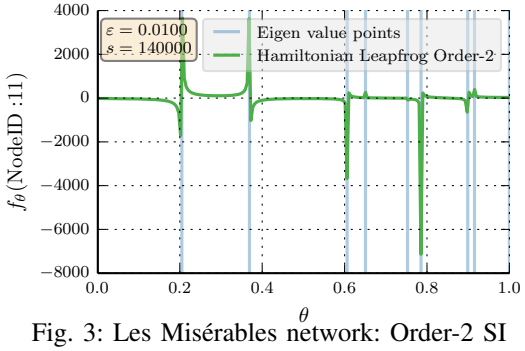


Fig. 3: Les Misérables network: Order-2 SI

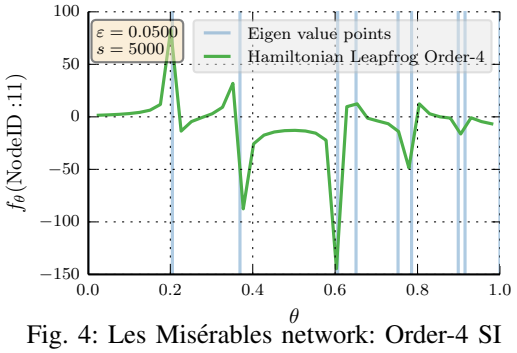


Fig. 4: Les Misérables network: Order-4 SI

B. Coauthorship graph in network science

The coauthorship graph represents a collaborative network of scientists working in network science as compiled by M. Newman [13]. The numerical experiments are done on the largest connected component with $n = 379$ and $m = 914$. Figure 5 displays the order-4 SI simulation and it can be seen that even though the eigenvalues are very close, the algorithm is able to distinguish them clearly.

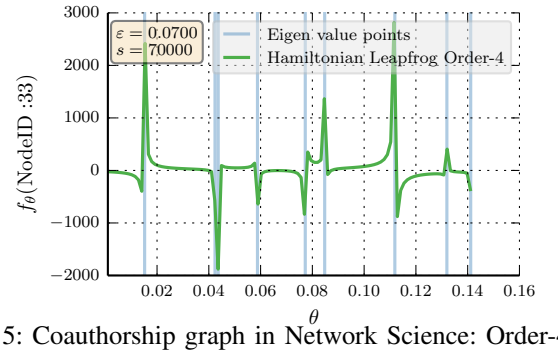


Fig. 5: Coauthorship graph in Network Science: Order-4 SI

C. Enron email network

The nodes in this network are the email ID's of the employees in a company called Enron and the edges are formed when two employees communicated through email[1]. Since the graph is not connected, we take the largest connected component with $33,696$ nodes and $180,811$ edges. The standard MATLAB procedures for eigenelements computation have difficulty to cope with such network sizes. The node under focus is the highest degree node in that component. Simulation result is shown in Figure 6.

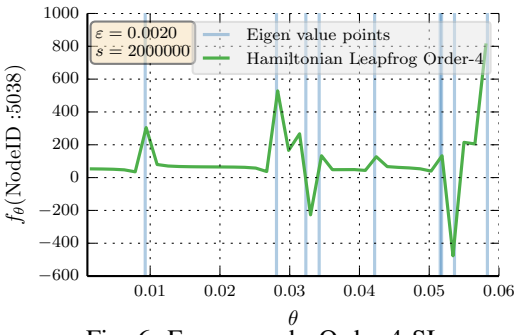


Fig. 6: Enron graph: Order-4 SI

VI. Applications

In this section we consider two related applications of the distributed spectrum computation to multi-agent and multi-dimensional (nD) systems.

First, we consider the consensus protocol in the wireless sensor networks [14]. We model a wireless sensor network as a random geometric graph, with nodes corresponding to agents (sensors) located on the unit square of the $\mathbb{R}^2$ plane and edges correspond to possible communication links within radius R. The consensus protocol is described as follows: let $x_k(t)$ be the value of sensor k at time slot t,

$$x_k(t+1) = \sum_{\ell \in N_{[k]}} w_{k\ell} x_\ell(t), \quad k = 1, ..., n,$$

or in the matrix form $x(t+1) = Wx(t)$, where $N_{[k]}$ is the set of neighbour nodes of node k including the node k itself, and $W = [w_{k\ell}]$ is a matrix of edge weights. Let the initial value

[1]Data collected from SNAP: http://snap.stanford.edu/data

of sensor k be $x_k(0) = m_k$. Then, if the consensus protocol converges, we have

$$\lim_{t\to\infty} x(t) = \bar{m}\mathbf{1}, \quad \bar{m} = \frac{1}{n}\sum_{k=1}^{n} m_k.$$

It has been demonstrated in [14] that performance of the best constant consensus protocol is very good in sparse wireless networks. The optimal weight value is given by [15]

$$w_{k\ell} = \frac{2}{\lambda_1(L) + \lambda_{n-1}(L)},$$

where $\lambda_t(L)$ denotes the t-th largest eigenvalue of the graph Laplacian $L = D - W$, $D = \text{diag}(W\mathbf{1})$. Using our distributed approach for the eigenvalues computation, we can propose a completely distributed, self-tuning, best constant consensus protocol.

We note that in the above example of the consensus protocol, the perfect consensus is actually reached only in the limit. Often, in practice we would like to obtain consensus in finite time. The finite-time consensus problem is very challenging and there is no simple solution for its design (see e.g., [16]). In [17] it has been suggested to use Iterative Learning Control (ICL) to achieve finite-time consensus. As in [17] we assume that each agent k can be described by a fairly general Markovian dynamics

$$y_k(t, r) = g_k(t) + h_k(q)u_k(t, r), \quad k = 1, ..., n,$$

where t indicates the time slots, whereas r indicates the ILC iterations. Here $g_k(t)$ is the zero input responsive function, $h_k(q)$ is the transfer operator with Markovian parameters and $u_k(t, r)$ is the control input.

The authors of [17] have proposed the following update rule for the control

$$u_k(t, r + 1) = u_k(t, r) + \gamma_k \sum_{\ell\in\mathcal{N}(k)} a_{k\ell}(y_\ell(T, r) - y_k(T, r)),$$

with $a_{k\ell}$ being elements of the graph adjacency matrix and γ_k being the learning gains to be designed, and shown that under such update rule, the system "learns" finite-time consensus, i.e.,

$$\lim_{r\to\infty} (y_\ell(T, r) - y_k(T, r)) = 0, \quad \forall k, \ell \in \{1, ..., n\},$$

if the following condition holds

$$\rho(I - H\Gamma L) < 1,$$

where $\Gamma = \text{diag}\{\gamma_1, ..., \gamma_n\}$ is the diagonal matrix of gains, $H = \text{diag}\{h_1(T), ..., h_n(T)\}$ is the diagonal matrix of the response function at time T and $\rho(A)$ is the spectral radius of matrix A. In fact, if the communication graph is undirected and connected, the condition $\rho(I - H\Gamma L) < 1$ is always satisfied. However, in practice, it is good to be not too aggressive in learning [18], and hence our distributed procedure can be used to choose gains $\{\gamma_k\}$ in such a way so that the value $\rho(I - H\Gamma L)$ estimated by our algorithm will be not too small and not too close to one.

VII. Conclusions and future research

We have proposed a distributed approach for the eigenvalue-eigenvector problem for graph matrices based on Hamiltonian dynamics, symplectic integrators and smoothed Fourier transform. We demonstrate with various network sizes that the proposed approach efficiently scales and finds, with higher resolution, closely situated eigenvalues and associated eigenvectors of graph matrices.

In the future we hope to present asynchronous versions of the introduced algorithms, to extend the proposed approaches to some classes of non-symmetric matrices and to design an automatic or a semi-automatic procedure for the identification of dominant eigenvalues and eigenvectors.

Acknowledgement

This work is supported by INRIA Bell Labs joint lab (ADR Network Science). We also would like to thank Leonid Freidovich for stimulating discussions. This is the author version of the IEEE nDS 2017 article.

References

[1] U. Von Luxburg, "A tutorial on spectral clustering," *Statistics and computing*, vol. 17, no. 4, pp. 395–416, 2007.

[2] C. Tsourakakis, "Fast counting of triangles in large real networks without counting: Algorithms and laws," in *IEEE ICDM*, Dec. 2008.

[3] D. Kempe and F. McSherry, "A decentralized algorithm for spectral analysis," in *STOC*, Jun. 2004.

[4] K. Avrachenkov, P. Jacquet, and J. K. Sreedharan, "Distributed spectral decomposition in networks by complex diffusion and quantum random walk," in *IEEE INFOCOM*, Apr. 2016.

[5] M. Franceschelli, A. Gasparri, A. Giua, and C. Seatzu, "Decentralized estimation of laplacian eigenvalues in multi-agent systems," *Automatica*, vol. 49, no. 4, pp. 1031–1036, 2013.

[6] T. Sahai, A. Speranzon, and A. Banaszuk, "Hearing the clusters of a graph: A distributed algorithm," *Automatica*, vol. 48, pp. 15–24, 2012.

[7] S. Blanes, F. Casas, and A. Murua, "Symplectic splitting operator methods for the time-dependent schrödinger equation," *The Journal of chemical physics*, vol. 124, no. 23, p. 234105, 2006.

[8] ——, "Splitting and composition methods in the numerical integration of differential equations," *arXiv preprint arXiv:0812.0377*, 2008.

[9] L. Verlet, "Computer" experiments" on classical fluids. i. thermodynamical properties of lennard-jones molecules," *Physical review*, vol. 159, no. 1, p. 98, 1967.

[10] B. Leimkuhler and S. Reich, *Simulating Hamiltonian Dynamics*. Cambridge University Press, 2004.

[11] A. Iserles, *A First Course in the Numerical Analysis of Differential Equations*, 2nd ed. Cambridge University Press, 2008.

[12] S. K. Gray and D. E. Manolopoulos, "Symplectic integrators tailored to the time-dependent schrödinger equation," *The Journal of chemical physics*, vol. 104, no. 18, pp. 7099–7112, 1996.

[13] M. E. J. Newman, "Finding community structure in networks using the eigenvectors of matrices," *Physical review E*, vol. 74, p. 036104, 2006.

[14] K. Avrachenkov, M. El Chamie, and G. Neglia, "A local average consensus algorithm for wireless sensor networks," in *IEEE DCOSS*, 2011, pp. 1–6.

[15] L. Xiao and S. Boyd, "Fast linear iterations for distributed averaging," *Systems & Control Letters*, vol. 53, no. 1, pp. 65–78, 2004.

[16] M. El Chamie, G. Neglia, and K. Avrachenkov, "Reducing communication overhead for average consensus," in *IEEE IFIP Networking*, 2013.

[17] D. Meng and Y. Jia, "Iterative learning approaches to design finite-time consensus protocols for multi-agent systems," *Systems & Control Letters*, vol. 61, no. 1, pp. 187–194, 2012.

[18] R. W. Longman and Y.-C. Huang, "The phenomenon of apparent convergence followed by divergence in learning and repetitive control," *Intelligent Automation & Soft Computing*, vol. 8, no. 2, pp. 107–128, 2002.